Búsqueda Dispersa Multiobjetivo de Promotores en Secuencias de ADN

R. Romero Zaliz[1], I. Zwir[2], F. Herrera[3]

[1] Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires, Argentina
email: rromero@dc.uba.ar

[2] Howard Hughes Medical Institute
Department of Molecular Microbiology
Washington University School of Medicine, USA
email: zwir@borcim.wustl.edu

[3] Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ing. Informática
Universidad de Granada, España
email: herrera@decsai.ugr.es

Resumen— **Uno de los problemas más tratados en el área de la *bioinformática*, área que abarca dos disciplinas: la biología y la computación, es el reconocimiento de *promotores de genes*. Encontrar buenas soluciones algorítmicas que permitan resolver este problema es muy importante para los investigadores del campo de la biología y la medicina ya que es un factor clave en el estudio de caminos (pathways) de regulación genética.**
En este trabajo presentaremos un enfoque multiobjetivo para resolver el problema de reconocimiento basado en el algoritmo clásico de *búsqueda dispersa*. Se aplicará el algoritmo a un conjunto de promotores de el organismo *Escherichia coli* y se compararán los resultados obtenidos con otros dos algoritmos multiobjetivos conocidos: el *Strength Pareto Evolutionary Algorithm (SPEA)* y el $(\mu + \lambda)$ *MultiObjective Algorithm*. [1]

Palabras clave— **Búsqueda Dispersa, Algoritmo Multiobjetivo, Reconocimiento de Patrones.**

I. Introducción

La identificación de la transcripción de genes es una pieza central en la maquina celular que controla el comportamiento y dinámica de la célula debido a que permite localizar la enzima, polimerasa o promotor, que participa en todos los procesos de regulación transcripcional. El descubrimiento de promotores resulta crucial para determinar cuales son los genes que participan de ese proceso de regulación y sus posibles conexiones en redes genéticas.
Particularmente, el reciente secuenciamiento de genomas y la sistematización de dicha información en bases de datos biológicas [1], [2] proveen una cantidad de información que facilita el reconocimiento de promotores en secuencias de ADN. Sin embargo, muchos métodos empleados en la predicción de los correspondientes sitios en secuencias de ADN presentan bajos índices de aciertos [3], [4], [5]. Esto se debe esencialmente a la dificultad de estos métodos en capturar la complejidad y vaguedad del modelo que describe la estructura de promotores en ADN. Por un lado, existen al menos tres submotivos con estructura propia separados por distancias variables. Por otro lado, los patrones son difusos en el sentido de presentar diversos errores o fallos.
En consecuencia, el problema de identificación de promotores constituye un problema de múltiples objetivos, donde se contraponen la calidad de la coincidencia (matching) de los submotivos en secuencias de ADN con las distancias adecuadas entre los mismos, y multimodal, donde resulta de interés descubrir todos los promotores optimales de una región regulatoria que permitan una verificación experimental del problema. En este trabajo proponemos un método que permite formalizar soluciones intuitivas e inexactas al problema de reconocimiento de promotores [5] proveyendo un marco computacional y de optimización a dicho problema.
Proponemos entonces un algoritmo de *Búsqueda Dispersa Multiobjetivo (MOSS)* basado en el algoritmo original de *Búsqueda Dispersa (SS)* [6] con algunas modificaciones para su adaptación a un ambiente multiobjetivo. Mostrare-

[1] Trabajo soportado por la RED HEUR TIC2002-10866-E y el Proyecto TIC2003-00877

mos el comportamiento de nuestra implementación en el problema de reconocimiento de patrones de promotores que será explicado en la Sección II. Luego, en la Sección III introduciremos el algoritmo propuesto detallando sus componentes e implementación. Compararemos el MOSS con otros algoritmos genéticos multiobjetivos (MOGAs) desarrollados recientemente: el *Strength Pareto Evolutionary Algorithm (SPEA)* y el $(\mu + \lambda)$ *MultiObjective Algorithm*. En la Sección IV se mostrarán los resultados de los experimentos y finalmente en la Sección V se realizará una conclusión sobre el trabajo realizado.

II. Reconocimiento de Patrones de Promotores

La investigación en el campo de la informática aplicada a problemas genómicos se está desarrollando rápidamente con avances en varios frentes. Los métodos para el reconocimiento y predicción de estructuras de genes proveen un punto clave para el análisis de genes y elementos funcionales de secuencias de ADN. El reconocimiento de genes involucra la identificación de elementos funcionales de ADN al igual que de señales reconocidas por la maquinaria de transcripción, montado y traducción de los organismos [7].

A. Promotores Procariotas

En general, los promotores procariotas comparten una estructura común identificada por patrones conservados. La gran mayoría de los promotores de *Escherichia coli (E.coli)* pertenecen a la clase de promotores σ^{70}, y tienen tres patrones conservados [8]:

1. *TTGACA-box:* Este patrón es una secuencia de seis pares de bases conservadas cuyo nucleótido central está localizado aproximadamente a 35 pares de bases río arriba del *Transcriptor Start Site (TSS)*, que es el sitio de inicio de la transcripción. La secuencia consenso de este patrón es `TTGACA`. Algunos caracteres están más conservados que otros, la relación es $\mathrm{T}_{82}\mathrm{T}_{84}\mathrm{G}_{78}\mathrm{A}_{65}\mathrm{C}_{54}\mathrm{A}_{45}$ [8] en donde, por ejemplo, la primera `T` es el nucleótido más visto en la primera posición del patrón y ha estado presente en 82 casos de cada 100.
2. *TATAAT-box:* Este patrón tiene también seis pares de bases y su nucleótido central está localizado aproximadamente a 10 pares de bases río arriba del TSS. La secuencia consenso es `TATAAT` y la relación en este patrón es $\mathrm{T}_{80}\mathrm{A}_{95}\mathrm{T}_{45}\mathrm{A}_{60}\mathrm{A}_{50}\mathrm{T}_{96}$ [8].
3. *Transcriptional Start Site (TSS):* En general, una pirimidina (`C` o `T`) seguida de una purina (`A` o `G`) componen el TSS de un gen. Esta secuencia indica el comienzo de la transcripción.

La distancia entre el TTGACA-box y el TATAAT-box es variable pero usualmente se encuentra entre los 16 y 18 pares de bases, y entre 15 y 21 pares de bases en el peor caso. Esta distancia puede ser crítica para la geometría de la ARN polimerasa [8].

B. Enfoque Computacional

Desde el punto de vista computacional, el problema biológico descrito tiene varias características interesantes. A pesar de que los promotores de *E. coli* son uno de los más estudiados dentro de los promotores procariotas, no existe una solución algorítmica perfecta para el problema de reconocimiento. Como se puede ver, existen tres patrones que pueden ser utilizados como modelos para la detección de promotores. Sin embargo, el tercer y último patrón, el TSS, es muy sencillo de encontrar. Esto es una desventaja ya que la gran mayoría de las instancias de este patrón detectadas son falsos positivos debido a la longitud que este patrón posee. Ésta es la razón por la cual algunos algoritmos deciden no utilizar este pequeño patrón como parte de la búsqueda, y es la misma razón por la cual nosotros tampoco lo utilizaremos.

Si realizamos una búsqueda de los patrones descritos en forma manual en una secuencia pequeña, encontraremos que existen varias soluciones posibles. Cada una de ellas es mejor a las otras en uno o más patrones y peor en otros. Por lo tanto, no es posible determinar cual de estas soluciones es la mejor. Este hecho convierte el problema de reconocimiento de patrones de promotores de genes en un problema *multiobjetivo*. Cabe destacar que también es un problema *multimodal* ya que existe más de una solución posible al problema presentado para cada patrón individualmente. A continuación introduciremos algunos de estos conceptos [9]:

Definición II.1: Un problema de optimización multiobjetivo se define como maximizar/minimizar $f(m)$ sabiendo que:

$$\begin{array}{ll} f_m(x), & m = 1, 2, \ldots, M; \\ g_j(x) \geq 0, & j = 1, 2, \ldots, J; \\ h_k(x) = 0, & k = 1, 2, \ldots, K; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}$$

en donde M corresponde al número de objetivos que tiene el problema, J al número de restricciones de desigualdad, K al número de restricciones de igualdad y, finalmente, n es el número de variables de decisión. Una solución x es un vector de n variables de decisión: $x = (x_1, x_2, \ldots, x_n)^T$.

El último conjunto de la Ecuación II.1 restringe cada variable de decisión x_i a tomar un valor con un mínimo $x_i^{(L)}$ y un máximo de $X_i^{(U)}$.

Definición II.2: Una solución x se dice que domina a una solución y ($x \prec y$), si las siguientes condiciones se cumplen:

- La solución x no es peor que y en todos los objetivos: $f_i(x) \not\triangleright f_i(y)$ para todo $i = 1, 2, \ldots, M$.
- La solución x es estrictamente mejor que y en, al menos, un objetivo: $f_j(x) \triangleleft f_j(y)$ para al menos un $i \in \{1, 2, \ldots, M\}$.

Si alguna de las condiciones anteriores es violada, la solución x no domina a la solución y. Si x domina a la solución y también es común escribir que x es *no-dominada* por y. Para poder codificar el algoritmo se utilizaron tres funciones objetivo, una para cada objetivo: TTGACA-box, TATAAT-box y la distancia entre estos dos patrones. Este último objetivo es esencial si se quieren obtener soluciones reales y útiles.

La función de aptitud de ambos patrones se han calculado de la misma manera, utilizando la información de la frecuencia consenso de los nucleótidos. El valor de esta función estará entre 0 y 1, cero para la peor y uno para la mejor adaptación al medio.

$$f_1 = f_2 = \frac{amount - min}{max - min} \tag{1}$$

en donde *amount* es calculado como la suma de la frecuencia obtenida para cada nucleótido observado.

$$amount = \sum_{i=1}^{6} cost(observedseq_i, knownseq_i) \tag{2}$$

min y max son calculadas como:

$$max = \frac{\sum_{i=1}^{6} knownfrec_i}{100} \tag{3}$$

$$max = \frac{\sum_{i=1}^{6} (100 - knownfrec_i)/3}{100} \tag{4}$$

en donde $observedseq_i$ es el nucleótido observado en la posición i y $knownseq_i$ es el nucleótido que es más frecuentemente observado en la posición i. La función *cost* es calculada como:

$$\begin{cases} knownfrec_i/100 & a = b \\ (100 - knownfrec_i)/300 & a \neq b \end{cases} \tag{5}$$

La distancia entre TTGACA-box y TATAAT-box es medida como una función triangular centrada en 17 en donde toma el valor máximo (uno), y a la derecha e izquierda de este punto el valor de la función decrece como puede verse en la Ecuación 6 y la Figura 1.

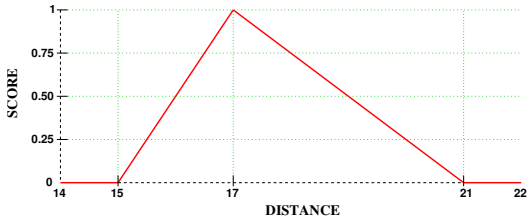

Fig. 1. f_3 gráficamente

$$f_3 = \begin{cases} -0{,}25 * distance + 5{,}25 & 17 \leq distance < 21 \\ 0{,}5 * distance - 7{,}5 & 15 < distance < 17 \\ 0 & sino \end{cases} \tag{6}$$

III. Algoritmo de Búsqueda Multiobjetivo

Para poder resolver el problema presentado utilizaremos una modificación del algoritmo original de búsqueda dispersa para trabajar con más de una función objetivo. Antes de comentar las modificaciones necesarias, realizaremos un breve introducción en la próxima subsección.

A. Búsqueda Dispersa

El algoritmo de búsqueda dispersa es una metaheurística basada en la idea de combinar las soluciones guardadas en un conjunto conocido como *Conjunto de Referencia* y aplicar búsqueda local a los individuos resultantes. Este conjunto acumula las *buenas* soluciones encontradas durante el proceso de búsqueda. Es importante notar que estas *buenas* soluciones no son solo aquellas buenas desde el punto de vista de la calidad, sino también desde el punto de vista de la diversidad.

La estructura básica de un SS genérico se muestra en la Figura III-A.

B. Enfoque Multiobjetivo

Para definir el problema de optimización multiobjetivo instanciaremos la Definición II.1.

- $M = 3$. Tenemos tres objetivos: f_1 y f_2 son las funciones objetivos para cada uno de los patrones, f_1 para TTGACA-box y f_2 para TATAAT-box. f_3 corresponde a la distancia entre estos dos patrones (recordemos las Ecuaciones 1 y 6).
- $J = 1$. Tenemos solamente una restricción de esta clase, g_1: la distancia entre los patrones no puede ser menor que 15 ó mayor que 21 pares de bases.
- $K = 0$. No existen restricciones de igualdad.
- Solamente se mantienen soluciones válidas en cada generación.
- Los patrones no pueden estar localizados fuera de los márgenes de la secuencia en donde se

1: Comenzar con $P = \emptyset$. Utilizar el método de generación para construir una solución y aplicar el método de búsqueda local para mejorarlo (x). Si $x \notin P$ entonces agregar x al conjunto P, sino, rechazar x. Repetir el procedimiento hasta que P tenga el tamaño especificado por el usuario.
2: Crear un conjunto de referencia $RefSet$ con las mejores $b/2$ soluciones de P y las $b/2$ soluciones más diversas de las primeras $b/2$.
3: Evaluar las soluciones en $RefSet$ y ordenarlas con respecto a su función objetivo.
4: $NewSolution \leftarrow$ `true`
5: **while** $NewSolution$ **do**
6: $NewSolution \leftarrow$ `false`
7: Generar subconjuntos de $RefSet$ en donde haya al menos una buena solución en cada uno.
8: **while** subconjunto a examinar **do**
9: Selecciona un subconjunto y lo marca como examinado.
10: Aplica la operación de combinación a las soluciones en el conjunto.
11: Aplica la operación de búsqueda local a cada solución nueva resultante de la combinación anterior.
12: **if** $f(x) < f(x^b)$ y $x \notin P$ **then**
13: $x^b \leftarrow x$
14: Reordenar $RefSet$.
15: **end if**
15: $NewSolution \leftarrow$ `true`.
16: **end while**
17: **end while**

Fig. 2. Algoritmo básico de búsqueda dispersa

está realizando la búsqueda.
En las próximas subsecciones algunas de las funciones y procedimientos más relevantes serán explicadas en detalle.

B.1 Componentes

Para la representación de cada individuo utilizamos una representación de bloques. Cada bloque corresponde a un patrones en el promotor, luego, tendremos dos bloques por individuo, uno para el TATAAT-box y otro para el TTGACA-box. Un bloque es representado por dos números enteros que definen un intervalo en donde el patrón específico está localizado en la secuencia en la cual se está buscando (vea la Figura 3 como ejemplo). El primer número entero representa el nucleótido inicial en donde se encuentra la secuencia consenso en la secuencia objetivo, mientras que el segundo número representa el tamaño del bloque.

Fenotipo

```
    ttgaca                  tataat
gtttatttaatgtttacccccataaccacataatcgcgttacact
```

Genotipo

Gen 0
`[(6,6)]`
Costo = 0.604651

Gen 1
`[(29,6)]`
Costo = 0.800725
Tamaño = 1.000000

Fig. 3. Ejemplo de representación

El proceso de combinación consiste en un *Cruce*. Este cruce es implementado como un operador de cruce simple de un punto, en donde el punto está siempre ubicado en medio de los dos bloques que forman el cromosoma. Los cromosomas tiene dos bloques A y B, entonces los padres A_1B_1 y A_2B_2 producirán como descendencia a A_1B_2 y a A_2B_1.

La *búsqueda local* implementada para este problema en particular consiste en una búsqueda entre soluciones vecinas. Cuando un cromosoma realiza una búsqueda local, se muta un número específico de nucleótidos a derecha e izquierda de uno de los bloques que conforman el cromosoma. Si el nuevo cromosoma domina a su padre, entonces el viejo cromosoma es reemplazado por el nuevo. Sino, el nuevo cromosoma es comparado con la población de individuos no-dominados encontrados hasta el momento, y si es no es dominado por ninguno de ellos, entonces reemplaza a su padre con una probabilidad del 50 % como se muestra en la Figura III-B.2.

B.2 Implementación

Alguno de los problemas más comunes de los algoritmos multiobjetivos son: la multimodalidad, la decepción, los óptimos aislados y el ruido colateral [9]. Este problema de reconocimiento de promotores en particular es afectado mayoritariamente por la multimodalidad. Es por ello que será necesario no solo mantener un conjunto de soluciones de buena calidad, sino también soluciones que mantengan la diversidad.

Para hacer que el algoritmo original de búsqueda dispersa funcione con más de un objetivo es necesario utilizar el concepto de *no-dominancia* [9] como fue mencionado en la Sección II-B.
En el algoritmo original de un solo objetivo, uno de los pasos involucrados es el ordenamiento de

la soluciones por la función de aptitud. Ahora tenemos más de un objetivo, con lo cual no podemos decir que una solución sea mejor que otra (no-dominancia). Para solucionar este problema, dividimos las soluciones en dos conjuntos, dominados y no-dominados. Luego, las soluciones no-dominadas se agregan al conjunto en cualquier orden, mientras que las soluciones dominadas se agregan al final si no hay más soluciones no-dominadas que incorporar, nuevamente en cualquier orden.
Un conjunto de individuos no-dominados es también utilizado para almacenar todas aquellas soluciones no-dominadas que han sido encontradas durante el proceso de búsqueda.
La Figura III-B.2 muestra el algoritmo MOSS.

IV. Experimentos

Para comparar los resultados obtenidos por el MOSS propuesto en este trabajo, aplicaremos tanto a éste algoritmo como a otros dos algoritmos de optimización: Strength Pareto Evolutionary Algorithm (SPEA) [10] y $(\mu + \lambda)$ Multi-Objective Evolutionary Algorithm (MuLambda)[11] el mismo problema ya comentado en la Sección II.
Los tres algoritmos comparten las siguientes propiedades:
- Acumulan las soluciones óptimas encontradas durante la búsqueda en un conjunto externo.
- Utilizan el concepto de dominancia de Pareto para asignar valores de aptitud a los individuos.

SPEA es un algoritmo metaheurístico bien conocido que tiene las siguientes propiedades [10]:
- Combina las técnicas anteriores en un solo algoritmo.
- La función de aptitud de un individuo es determinada por las soluciones acumuladas en el conjunto externo, la dominancia con respecto a los individuos dentro de la misma población es irrelevante.
- Todas las soluciones del conjunto externo participan en el proceso de selección.
- Un método de nichos es provisto para poder preservar la diversidad en la población, este método está basado en la optimalidad de Pareto y no requiere un parámetro de distancia (como el radio del nicho en *sharing* [9]).

MuLambda es un algoritmo relativamente nuevo y tiene un diseño muy diferente a los demás enfoques basados en Pareto. Este algoritmo tiene las siguientes propiedades [11]:
- No utiliza ninguna información de los individuos dominados de la población, solamente los individuos no-dominadas se mantienen de generación en generación.
- El tamaño de la población es variable.

1: Comenzar con $P = \emptyset$. Utilizar el método de generación para construir una solución y el método de búsqueda local para mejorarla (x). Si $x \notin P$ entonces agregar x a P, sino, rechazar x. Repetir el procedimiento hasta que P tenga el tamaño definido por el usuario.
2: Crear un conjunto de referencia $RefSet$ con $b/2$ soluciones no-dominadas de P y $b/2$ soluciones de P que resulten diversas de las primeras $b/2$. Si no existe suficiente cantidad de soluciones no-dominadas para llenar el conjunto, completar con soluciones dominadas en cualquier orden.
3: $NewSolution \leftarrow$ `true`
4: **while** $NewSolution$ **do**
5: $NewSolution \leftarrow$ `false`
6: Generar subconjuntos de $RefSet$ en donde exista al menos una solución no-dominada en cada uno.
7: Generar un conjunto vacío N para acumular las soluciones no-dominadas.
8: **while** subconjunto para examinar **do**
9: Seleccionar un subconjunto y marcarlo como examinado.
10: Aplicar el operador de combinación a las soluciones del conjunto.
11: Aplicar el operador de búsqueda local para cada nueva solución encontrada luego de aplicarle el operador de combinación como se explica en la Figura Figure III-B.2.
12: **if** x^b no es dominado por ningún $x \in N$ y $x \notin N$ **then**
13: Agregar x^b a N.
14: **end if**
15: **end while**
16: Agregar las soluciones de N a P si no son dominadas por ningún integrante de P.
16: $NewSolution \leftarrow$ `true`.
17: **end while**

Fig. 4. Algoritmo MOSS

- Realizar *agrupamientos (clustering)* para reducir el número de soluciones no-dominadas acumuladas sin destruir las características del frente óptimo de Pareto.

Como se ha explicado en las secciones previas, el MOSS tiene las siguientes características propias:
- Utiliza búsqueda local para mejorar las soluciones encontradas hasta el momento durante la ejecución del algoritmo.
- Mantiene la diversidad de las soluciones incluyendo soluciones diversas a las que ya han sido encontradas.

Los tres algoritmos utilizan las mismas funciones objetivo descriptas en la Sección III-B y han sido

1: Seleccione aleatoriamente que gen de g será mutado en el cromosoma seleccionado c.
2: Seleccione aleatoriamente un número n en $[-vecino, vecino]$ y mueva el gen g, n nucleótidos. Dado que el número n puede ser negativo, la posición del patrón puede moverse tanto río arriba como río abajo. El gen resultante será g' y el cromosoma resultante será llamado c'.
3: **if** c' cumple las restricciones **then**
4: **if** c' domina a c **then**
5: Reemplazar c con c'.
6: **end if**
7: **if** c' no domina a c y c' no es dominado por c y c' no es dominado por ninguna solución en el conjunto externo de soluciones no dominadas **then**
8: Reemplazar c con c' con un 50 % de probabilidad.
9: **end if**
10: **end if**

Fig. 5. Búsqueda local

aplicados a un conjunto de secuencias de promotores conocidas reportados en la literatura [1]. En este trabajo, se muestran 272 secuencias de promotores y las posiciones en donde se encuentran los distintos patrones. También se incluyeron 78 localizaciones alternativas a estos patrones en las mismas secuencias. Nuestro objetivo es encontrar no solo los promotores sino también las localizaciones alternativas para todas las secuencias.
Para cada una de las secuencias se utilizaron 20 semillas diferentes generando 20 conjuntos de soluciones diferentes. Un promotor se dice que ha sido encontrado si aparece en, al menos, uno de estos conjuntos.
Para comparar los diferentes algoritmos, dos funciones C[12] y D (ver Ecuaciones 7 y 8) fueron utilizadas. El valor $C(X', X'') = 1$ significa que todas las soluciones en X'' son dominadas o son iguales a las soluciones en X'. Su opuesto, $C(X', X'') = 0$, representa la situación en donde ninguna de las soluciones en X'' fueron cubiertas por las soluciones de X'. Es importante notar que tanto $C(X', X'')$ como $C(X'', X')$ deben ser consideradas, ya que $C(X', X'')$ no es necesariamente igual a $1 - C(X'', X')$.
La función $D(X', X'')$ cuenta la cantidad de individuos de X' que no dominan a X'' que no han sido encontrados por X''.

Definición IV.1: Sean $X', X'' \subseteq X$ dos conjuntos de vectores de decisión. La función C mapea los pares ordenados (X', X'') al intervalo $[0, 1]$:

$$C(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a' \preceq a''\}|}{|X''|} \quad (7)$$

$$D(X', X'') = |\{a' \in X'; a'' \in X'' : a'' \not\preceq a' \wedge a' \neq a''\}| \quad (8)$$

A. Resultados

En la Tabla I se muestra la cantidad de promotores detectada por cada algoritmo, tanto de los patrones originales como de las alternativas válidas para algunos patrones según se informa en [1]. Como se puede apreciar se ha logrado obtener una mejora con respecto a los otros dos enfoques multiobjetivos llegando a superar el 90 % de aciertos.
En la Tabla III se presentan los resultados promedios para las secuencias de promotores seleccionadas para la experimentación. En esta figura se pueden ver dos tablas, una para $C(X', X'')$ y otra para $D(X', X'')$. Estos números han sido obtenidos ejecutando el algoritmo 20 veces con diferentes semillas y calculando el valor promedio para cada función y para todas las secuencias.
Cada algoritmo ha sido ejecutado con los parámetros de la Tabla II.

Parámetro	Valor
Número de generaciones	200
Tamaño de la población	8
Tamaño de la población de no-dominados	300

TABLA II
PARÁMETROS PARA LOS ALGORITMOS

$C(X', X'')$	MOSS	spea	$\mu + \lambda$
MOSS	0.140	0.538	0.360
spea	0.013	0.330	0.054
$\mu + \lambda$	0.029	0.349	0.111

$D(X', X'')$	MOSS	spea	$\mu + \lambda$
MOSS	1.907	14.204	12.977
spea	0.170	1.077	0.876
$\mu + \lambda$	1.066	2.284	1.337

TABLA III
RESULTADOS PARA LAS SECUENCIAS

Los resultados son buenos para el enfoque MOSS y no se presentan fluctuación muy grandes entre las diferentes secuencias. MOSS es el algoritmo que lidera las tablas seguido por el MuLambda y SPEA al final en todos los casos.
Es muy importante notar que la diversidad de las soluciones encontradas por el MOSS con el mismo número de ciclos de población es mejor que la encontrada por los demás algoritmos comparados.

	Originales	Alternativas	%originales	%alternativas	Total	%total
MOSS	252	65	92,65 %	83,33 %	317	90,57 %
SPEA	222	48	81,62 %	61,54 %	270	77,14 %
MuLambda	230	58	84,56 %	74,36 %	288	82,29 %

TABLA I
RESULTADOS PARA LAS SECUENCIAS DE PROMOTORES

Como ya hemos dicho, la función D cuenta el número de individuos no-dominados de un experimento que no han sido encontrados en los demás. MOSS logra obtener el mejor valor en todos los experimentos, mientras que SPEA y MuLambda tienen valor notablemente menores.
El tamaño máximo de la población externa de no-dominados es de 300 individuos, suficientemente grande para contener a todos los individuos encontrados durante la ejecución. MOSS tiene más de siete veces más diversidad que los otros dos algoritmos.

Es también importante notar que, en promedio, SPEA encuentra el promotor en 6,48 de las 20 ejecuciones, MuLambda en 9,33 y MOSS en 16,81 haciendo a nuestro enfoque el más robusto y confiable.

V. Conclusiones y Trabajo a Futuro

Desde el punto de vista del problema biológico hemos planteado un método que permite formalizar soluciones intuitivas e inexactas del problema de reconocimiento de promotores proveyendo un marco computacional y de optimización a dicho problema.
Desde el punto de vista metodológico hemos propuesto una versión multiobjetivo del clásico algoritmo de *búsqueda dispersa* y utilizado para detectar los promotores de la clase σ^{70} de E.coli. Los resultados muestran que nuestro enfoque es competitivo y presenta mejores resultados que los otros dos algoritmos multiobjetivos comparados: Strength Pareto Evolutionary Algorithm y $(\mu+\lambda)$ Multi-Objective Evolutionary Algorithm.

En secuencias de ADN muy extensas el espacio de búsqueda es muy grande y puede perjudicar a un algoritmo genético multiobjetivo. Es por ello que se estudiará a futuro la hibridación de esta metodología con un análisis del espacio de búsqueda con *Redes Neuronales* [3] para facilitar el trabajo de nuestro algoritmo multiobjetivo.

Referencias

[1] W. R. McClure D. K. Hawley, "Compilation and analysis of escherichia coli promoter dna sequences," *Nucleic Acids Res.*, vol. 11, pp. 2237–2255, 1983.

[2] H. Salgado et al., "Regulondb (version 3.2): transcriptional regulation and operon organisation in escherichia coli k-12," *Nucleic Acids Research*, vol. 29, pp. 72–74, 2001.

[3] M. G. Reese, "Application of a time-delay neural network to promoter annotation in the drosophila melanogaster genome," *Computers & Chemistry*, vol. 26, no. 1, pp. 51–56, 2002.

[4] G. Stormo A. Ulyanov, "Multi-alphabet consensus algorithm for identification of low specificity protein-dna interactions," *Nucleic Acids Research*, vol. 23, no. 8, pp. 1434–1440, 1995.

[5] J. Collado-Vides A. M. Huerta, "Sigma70 promoters in escherichia coli: specific transcription in dense regions of overlapping promoter-like signals," *J Mol Biol.*, vol. 333, no. 2, pp. 261–278, 2003.

[6] R. Martí M. Laguna, *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, Boston, 2003.

[7] J. W. McLarty C. H. Wu, *Neural Networks and Genome Informatics*, Elsevier, 2000.

[8] W. McClure D. Hawley, "Compilation and analysis of escherichia coli promoter dna sequences," *Nucleic Acids Res*, vol. 11, pp. 2237–2255, 1983.

[9] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[10] L. Thiele E. Zitzler, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," in *Technical Report TIK-Report 43*, May 1998.

[11] C.Ñewton R. Sarker, K. Liang, "A new multiobjective evolutionary algorithm," *European Journal of Operational Research*, vol. 140, pp. 12–23, 2002.

[12] L. Thiele E. Zizler, K. Deb, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[13] R. Beausoleil, "Multiple criteria scatter search," in *MIC2001 - 4th Metaheuristics International Conference, Porto, Portugal*, 2001, pp. 539–543.